METABOLOMICS

Teresa Emilea Norris

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Science

Department of Mathematics and Statistics

University of North Carolina Wilmington

2006

Approved by

Advisory Committee

Dr. James Blum Dr. Dargan Frierson

Dr. Susan Simmons
Chair

Accepted by

Dr. Robert Roer
Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

Metabolomics is the newest of the "-omics" sciences showing great potential in identifying biomarkers for drug discovery. Since Metabolomics is a relatively new science there are a few issues that have not been investigated to a great extent. As advances in this area are being made, there is a need for better analysis methods that provide adequate and trustworthy results.

Presented in this research are the issues that cloud the great potential for analyzing a metabolomics data set. One of the biggest issues comes from the missing values of metabolite concentrations due to an assay threshold. Different imputation methods are evaluated, and how these affect feature selection with respect to a disease status.

The goal is to provide more research in these areas through exploration of imputation and feature selection methods. Once conclusions are made, there will be a pathway to help further metabolomics research.

ACKNOWLEDGMENTS

I would like to thank all of my professors at the Mathematical and Statistics Department of the University of North Carolina Wilmington for their guidance and enthusiasm for both fields, requiring me to be a well rounded student and professional. My greatest recognition, to my thesis advisor Dr. Susan Simmons who has shown me numerous ways to expand and explore topics that help incorporate statistics in new areas. Dr. James Blum and Dr. Dargan Frierson have not only served as advisors for this thesis, but as dedicated educators throughout my years at UNCW.

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Metabolomics is the newest of the "-omics" sciences showing great potential in identifying potential biomarkers for drug discovery. Metabolomics involves the bio-chemical profiling of all the metabolites in a cell, tissue, or organism (National Cancer Institute, 2005). This relatively new science focuses on the best measurement of the physiological state of organism's metabolites (Schmidt, 2004). Although metabolomics has the possibility of providing insight to many biological questions, data produced in metabolomics experiments pose a number of statistical challenges. New and sophisticated methods are needed to analyze metabolites and metabolomic profiles.

The areas of the "-omics" sciences focus on different levels of the human biological system. Genes, proteins, and metabolites are just a few of the active parts of this system. The study of gene and protein expressions within samples is known as genomics and proteomics, respectively; metabolomics, in a similar fashion, is concerned with metabolites, which are all the small molecules present in an organism. Some familiar examples of metabolites include ATP, glucose, and amino acids. One goal in metabolomics is to discover which cellular processes have been altered in a diseased individual, and identify potential drug therapies (Schmidt, 2004). These cellular processes are well known and an illustration of the Amino sugar process (Kegg, 2006) is shown in Figure 1.

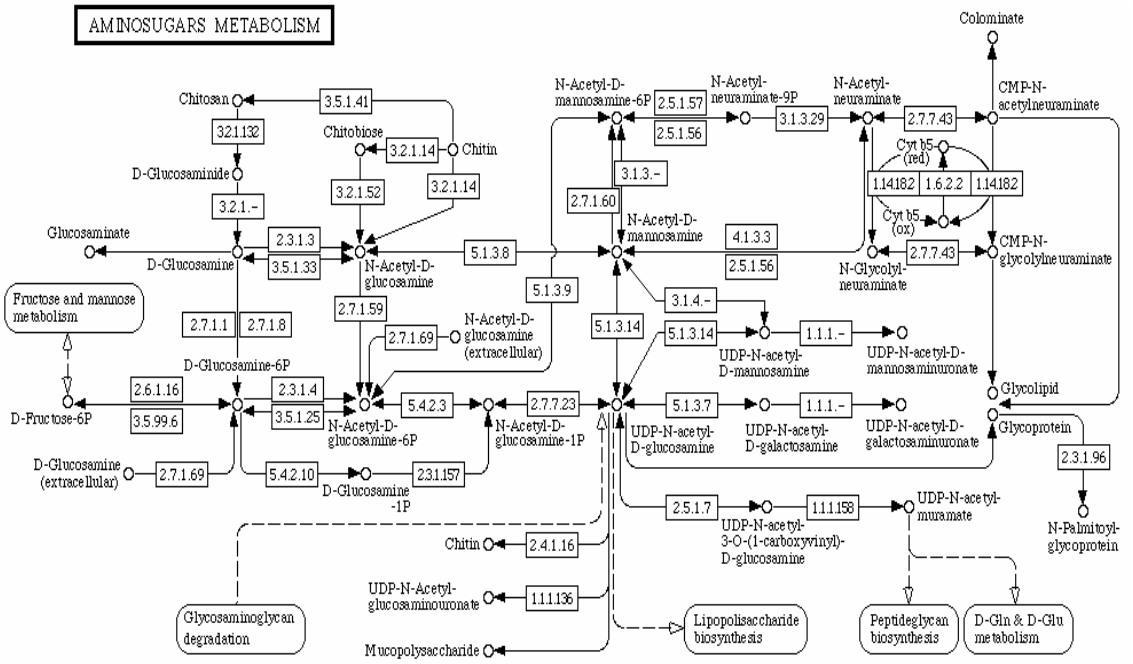

Figure 1. Cellular process of Amino sugar.

How will metabolites found in the human body ever be helpful in indicating if there is a current or developing disease or another imbalance? Currently in use are a few tests that have been in use for a number of years based on individual metabolite levels. For example, the test to diagnose diabetes includes seeking elevated levels of the metabolite glucose. Another popular disease biomarker is cholesterol, in which high levels are associated with heart disease and strokes (Schmidt, 2004). The search for more featured metabolites will produce just as helpful indicators and ultimate therapies in other conditions.

Another drive for furthering the metabolomics research comes from hopes that not only seeking single metabolites, but associated groups will shed light on actual causes for disease(s). For example, in the case of high cholesterol levels, identifying a group of metabolites associated with this condition could pin point which cellular processes are altered. These metabolomic profiles can give more information on what to do to prevent or alleviate the problem, besides just

giving potential treatment for the disease. Imagine being able to gain information on the underlying biological causes of any disease.

A number of challenges populate metabolomics data, including non-normal distributions of metabolites, correlations among metabolites, and missing values. These datasets also have dimensions where the number of biological samples is much smaller than the number of metabolites measured with in each sample ($n < p$ problem for n samples and p metabolites). There are estimated to be at least 2,500 metabolites in the human body (Beecher, 2005). This paper will address some of these statistical challenges, with the goal of presenting appropriate methods to overcome these issues.

One of the biggest statistical issues in metabolomics is dealing with missing values. In most human metabolomics experiments, hundreds of metabolites are measured through Mass Spectrometry (MS) or Nuclear Magnetic Resonance Spectroscopy (NMR). However, these machines can only detect concentrations above a certain threshold, and this creates many missing values for a number of metabolites. Thus missing values are not generally missing at random; in fact, most missing values are due to concentrations below detection limits on the MS machines. How scientists deal with these missing values is a question that is not often addressed in the metabolomics literature. Common methods include replacing the missing value with the lowest concentration observed for a given metabolite (or the observed minimum), or by half of this minimum value, or 0 (personal communication, Dr. Matthew Mitchell). More recently, other methods such as random forests (which has its own imputation algorithm) have been used (Beecher, et. al., 2004). This thesis will address the issue of missing values in metabolomics data and explore the impact each has on the analysis of the data.

Several analysis methods are investigated herein, including Random Forests, t-test, Partial Least Squares, and Robust Singular Value Decomposition. The primary objective is to understand which analysis method is best for identifying metabolites that differentiate between two response groups, say, diseased versus healthy. These methods are described in Section 2.

Continuing into Section 3, the 5 different imputation methods under investigation will be explained. In Section 4, the metabolomics data sets and simulation methods will be described. Simulations will be created to explore the results under each scenario. Evaluations on the methods will be made by the results presented in Section 5. The "best" imputation and analysis methods will be determined and discussed in Section 6. Section 7 will discuss the conclusions and the next steps for further analysis.

ANALYSIS METHODS

Random Forest

The Random Forests algorithm was developed and studied by the late Leo Breiman, whose method has gained more popularity in a number of fields involving feature selection. Here, feature selection was used to find metabolites (the sought features) that are important markers for a certain disease. The basis of the Random Forest classification algorithm is growing a "forest" of many classification trees (Breiman, 2001). The goal of a single classification tree is to predict membership of cases in the classes of a categorical dependent variable from the measurements on one or more predictor variables. In one tree of a random forest, the classification of a case is obtained by sending the case down the constructed tree, and observing the class for which that tree "votes". These votes are summed up from every tree in the forest that casts a vote for a case. The class with the most votes will be the predicted class for the case.

A classification tree in the forest is grown in the following manner:

Step 1: All data are represented in the root node.

Step 2: Select the "best feature" of the randomly chosen "m_{try}" variables that separates the data. (See discussion below.)

Step 3: If the daughter node has all of the same class represented, then this is a leaf or an end node. Otherwise, repeat Step 2.

Step 4: Stop growing the tree if there are no remaining predictors available or samples with-in the daughter nodes belong to one case.

In Step 1, two thirds of the cases are randomly sampled with no replacement to construct a classification tree. The "m_{try}" option in random forest sets how many variables will be tried at each split of a node. The value of m_{try} by default is set to $\sqrt{p}$, where p is the number of predictor variables, or the user can define. In Step 2, the node will split on the median of the "best" of m_{try} randomly selected variables. Of the m_{try} variables, if there are no "best" features, then one of these variables is randomly selected and a random value is selected to split on this variable.

An illustration of a classification tree from Random Forest is represented in Table 1 and Figure 2. There are 13 nodes in this tree with disease classes 1 and 2. Node 1 is the root node, containing all observations as discussed in Step 1 above. The Left and Right Daughter nodes are created when there is a split on the best feature of the m_{try} variables at that node (or a weak variable randomly selected). The Status in Table 1 indicates if the node is going to have a daughter node (1) or stop with a classification (-1). A Prediction occurs at the terminal nodes, in this case at Nodes 5, 8, 9, 10, 11, 12, and 13. For example, Node 2 split by variable 283. Observations that have values less than 874.9845 for variable 283 go into node 4. Observations that have values greater go into node 5. Notice in this example, node 4 continues on with

daughter nodes, but node 5 is a terminal node that classifies observations into class 2. This is just one classification tree from the forest.

Node	Left Daughter	Right Daughter	Split Variable	Split Point	Status	Prediction
1	2	3	39	913.7295	1	0
2	4	5	283	874.9845	1	0
3	6	7	99	769.6994	1	0
4	8	9	83	1068.537	1	0
5	0	0	0	0	-1	2
6	10	11	41	877.2128	1	0
7	12	13	258	677.2041	1	0
8	0	0	0	0	-1	2
9	0	0	0	0	-1	1
10	0	0	0	0	-1	2
11	0	0	0	0	-1	1
12	0	0	0	0	-1	1
13	0	0	0	0	-1	2

Table 1. A Tree in a Random Forest run.

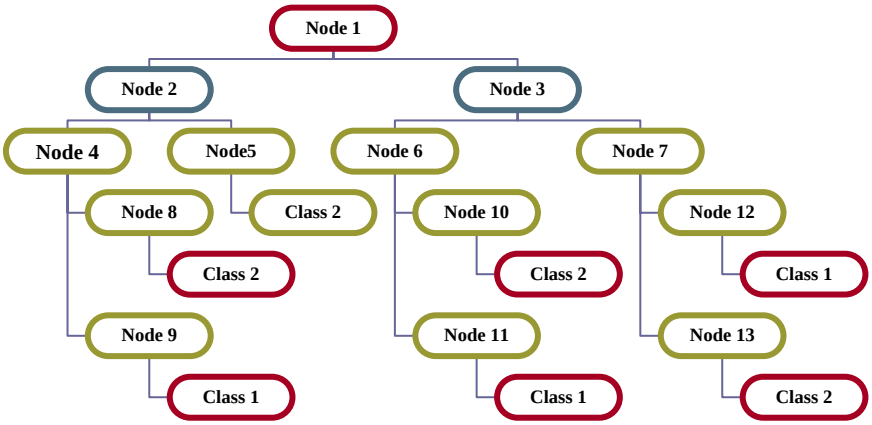

Figure 2. Layout of Tree from Table 1.

Random Forest uses the algorithm of classification trees and grows a "forest" of these trees. This forest is constructed by creating a number of classification trees based on the steps mentioned above. By default, 500 trees have been thought to be sufficient, but can be defined by the user. Growing many trees instead of just a single tree, results in significant improvements in the classification accuracy (Breiman, 2001).

Noting that two thirds randomly selected cases are used in the construction of each tree, the cases not used are coined as “out-of-bag” (oob) observations. These observations are used for feature selection (metabolites) and to get an unbiased estimate of the classification error. An oob observation for a tree is run down the tree and based on the splits of the tree, casts a “vote” for the class. Each observation is run down every tree in which it is considered oob, and every tree casts a “vote” for the classification of that observation. The class with the highest number of votes will be the classification for that observation. The oob error estimate is calculated by the proportion of all cases not correctly classified by the forest of trees.

The identification of the important features in a random forest (RF) is obtained by calculating the importance of each variable. The RF uses the oob observations of each tree and counts the number of votes for the correct class. The values the m^{th} variable from the tree are randomly permuted for the oob observations, put back down the tree again, and the number of votes for the correct class is counted. By subtracting the number of votes for the correct class in the permuted oob observations from the original oob observations and taking the average of this number over all trees in the forest gives the raw importance score for the variable is calculated. In the Random Forests Package for R, there is an option for importance listing and importance plot. Figure 3 illustrates what the importance plot output would look like. The plot shows elements V12 and V136 as having the highest importance values.

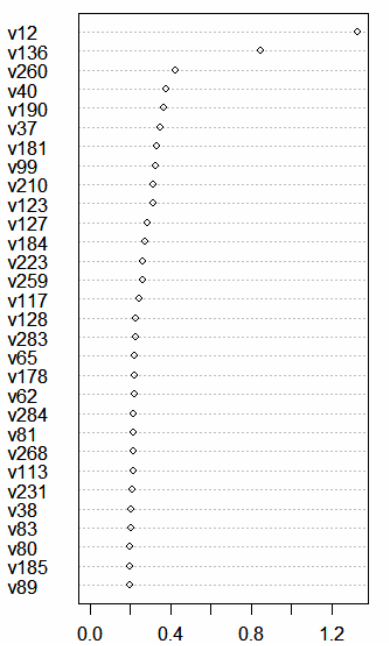

Figure 3. An Example of an Importance Plot

t-tests

In traditional terms, a t-test of no difference between two groups compares the observed difference between two group means in relation to the observed variation in the data. The t-test is used to measure the statistical significance amongst the two classes of diseases for each metabolite. The test statistics for testing no difference between two groups is

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$

For this research, the overall significance level was set a 0.05, which means the experiment-wise type I error rate is 0.05. A bonferroni adjustment was used so, the actual significance level for each metabolites t-test is at 0.05 divided by the number of metabolites. Table 2 shows an example of the t-test output. Here there were 205 predictor variables, so following the bonferroni adjustment, any predictors p-value would have to be below 0.05/205 = 0.0002 to be significant. As listed in Table 2, these all have significant p-values, therefore are significant factors.

Variable	tValue	P-value
V7	4.44	<.0001
V8	4.7	<.0001
V15	4.74	<.0001
V30	10.61	<.0001
V31	8.49	<.0001
V37	4.35	<.0001
V170	4.83	<.0001
V187	5.76	<.0001

Table 2. t-test Example

Partial Least Squares

Partial Least Squares Regression (PLS) constructs a predictive model given there are possible underlying relationships observed. When the problem of finding the relationship between the response and predictor variables is presented, multiple linear regression may come to mind as a good analysis technique (Tobias, 2006). This is true when predictors are present with a well-understood relationship to the response, and no significant collinearity amongst these predictive variables exist. Due to the nature of metabolomic data, multiple linear regression or logistic regression is most likely not appropriate. The number of metabolites tend to be much larger than the number of samples (n<p problem), which prevents fitting the full model. Also, the correlations among metabolites induce a good amount of multicolinearity in the model. PLS has an emphasis on predicting a response with underlying relationships between the predictor variables. PLS uses factors to predict the responses.

As in multiple linear regression, the main purpose of PLS is to build a linear model such that

$$y = X\beta + E$$

where y is the response vector of size n by 1, X is a n by p matrix of predictors, β consists of the coefficients for the predictors with size p by 1, and E is an residual matrix of size n by 1 (Friedman, et.al., 2001). For the PLS case, X is written as

$$\begin{aligned} X &= TC' + E_p \\ &= t_1 c_1' + t_2 c_2' + \cdots + t_p c_p' + E_p \end{aligned}$$

Where T=[t_1, t_2,..., t_p], C=[c_1, c_2, ..., c_p]. The t_p's are called latent variables or scores, and the c_p's are called loadings. The E_p is the residual matrix (unexplained part of the X scores) and k is the number of PLS components. Therefore, T has dimensions n by p, C is a p by p matrix, and E_p is n by p.

PLS "links" X and y by the latent variables T

$$\begin{aligned} y &= X\beta \\ &= TQ + f_p \\ &= t_1 q_1 + t_2 q_2 + \cdots + t_p q_p + f_p \end{aligned}$$

where Q=[q_1, q_2,..., q_p], is a p by 1 matrix consisting of regression coefficients or "loadings" for the response. Residual vector f_p, has size n by 1.

For the analysis in this paper, the X matrix was first standardized for each of the p metabolites present. The first component (c_1, the first loading) explains the most variability, or best approximation of X. The primary focus of this research was to seek important metabolites using c_1's 10 highest absolute values that correspond to the top 10 important metabolites.

Robust Singular Value Decomposition

Robust Singular Value Decomposition (rSVD) was recently introduced in a technical report from the National Institute of Statistical Sciences (Hawkins, et. al., 2001). rSVD was initially developed to analyze microarray data (Hawkins, et. al., 2001). Microarray data tend to have missing values, outliers, non-normal data, and an inherent correlation structure, many of the

same issues found with metabolomics data. Potentially, the rSVD algorithm will help overcome some of the challenges in analyzing metabolomics data.

To understand the foundation for rSVD creation, Singular Value Decomposition (SVD) needs to be defined in linear algebra terms. SVD can be thought of as a way to understand the structure of a rectangular matrix (n cases by p variables, $n \neq p$), allowing for insight on relationships between row (cases) and column (variables) factors. In the metabolomics case, the classification status has a noticeable relationship with metabolite(s). The SVD of an n by p data matrix X is decomposed into left and right eigenvectors (U,V) and eigenvalues (S) as follows

$$X = U_{(nxp)} S_{(pxp)} V^T_{(pxp)} .$$

Looking at the i^{th} left and right eigenvectors ($\boldsymbol{u}_i,\boldsymbol{v}_i$) and eigenvalue ($s_i$)

$$X = \sum_{i=1}^{p} s_i \boldsymbol{u}_i \boldsymbol{v}_i^T .$$

Note that another important feature of SVD comes from the k^{th} approximation for the SVD of X. Just using the first k terms in of a data matrix will be noted as

$$X \approx \sum_{i=1}^{k} s_i \boldsymbol{u}_i \boldsymbol{v}_i^T .$$

Starting with the first eigen-triplet ($\boldsymbol{u}_1$, s_1, and $\boldsymbol{v}_1$), most of the information is contained in the decomposition of X. As k is increased, more information is given about X, but not enough to significantly improve what was found by the first eigen-triplet. rSVD has used the idea by using the ordered first left and right eigenvectors.

This idea has been applied to number of unrelated topics such as image resolution and document issues, but in each case the data matrix can be represented as a full complete matrix. This method is useful for data classification and clustering. However, this definition of SVD

does not allow missing values in the data matrix, which is one of the problems for metabolomics data and other real situations.

The alternating Least Squares SVD method (Gabriel, 1979) is based on least squares techniques which are very sensitive to outlying data causing another draw back to alternating Least Squares. One cell in the *X* data matrix being sufficiently outlying can cause SVD to draw the leading principal component toward itself**.** This outlying observation will have a much stronger importance as the observation becomes more extreme. This is where rSVD has an advantage due to its "robustness".

Similar to the alternating Least Squares SVD steps, the rSVD algorithm is as follows:

Step 1: Start with an initial estimate of $\boldsymbol{u}_1$.

Step 2: For each column j fit the L1 regression coefficient c_j by

$$\min \sum_{i=1}^{n} | x_{ij} - c_j u_{i1} | .$$

Step 3: Calculate the resulting estimate of the right eigenvector $v_1 = \frac{c}{|| c ||}$,

where ||c|| is the Euclidean norm of c such that, $|| c ||= \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$.

Step 4: Use estimate of $\boldsymbol{v}_1$ (right eigenvectors) to refine estimate of $\boldsymbol{u}_1$ (left eigenvectors).

Step 5: For each row *i* fit the L1 regression coefficient d_i by

$$\min \sum_{j=1}^{p} | x_{ij} - d_i v_{j1} | .$$

Step 6: Calculate the resulting estimate of the left eigenvector $\boldsymbol{u}_1 = \frac{d}{|| d ||}$.

Step 7: Iterate to convergence.

Another question that may come to mind is where to the start the $\boldsymbol{u}_1$ vector. There is not a clearly defined choice for the initial left leading eigenvector. It is suggested to start at the absolute value of the row medians and rescaling to unit length.

In the analysis phase, the leading left and right eigenvectors ($\boldsymbol{u}_1$, $\boldsymbol{v}_1$) will be arranged from largest to smallest components. Looking at the ordered left components, each observation should be grouped together in a way such that they are considered "similar". Along the response, groupings of classes should appear. This is the same for the ordered right components, but now looking at the predictor variables. The results as presented should be an easily read data set, ordered into groupings as influenced in the underlying structure.

Looking at Tables 3 and 4, a small example is presented for illustration purposes consisting of the ordered values of both leading left and right eigenvector's five highest values. Table 3 shows for this data's response consisting of "A" and "B", there are groupings present "B".

Response	u_1
B	0.280731
B	0.196362
B	0.186600
B	0.181231
B	0.180876

Table 3. Leading Left Eigenvector ($\boldsymbol{u}_1$)

Predictors	v_1
M122	0.137875
M109	0.130424
M192	0.124618
M67	0.123654
M177	0.115819

Table 4. Leading Right Eigenvector ($\boldsymbol{v}_1$)

IMPUTATION METHODS

When analyzing any real life metabolomics data set challenges arise from the potential of missing values. One of the goals of this research is to find a good method to "fill-in" or impute missing values for metabolomics data. Five methods for imputation will be discussed below.

Since metabolomics data is looking at metabolite concentrations, a standard way of imputing missing values is to replace the missing value with 0. The assumption here is there are no concentrations available below the limit of quantification, which is most likely false. However, imputing values using zero is kept in this research for comparison. The second imputation strategy uses the minimum observed value within each metabolite (different metabolites have different threshold values). The third method is to use half of the minimum within each metabolite. This imputation method recognizes that if there is no observation at that point then it must be smaller than any measured value. The fourth strategy imputes the missing values by randomly drawing from a uniform(0, Θ_i) distribution, where Θ_i is the observed minimum within the i^{th} metabolite. The minimum, half-minimum, and uniform(0, Θ_i) consider there may still be a small amount of concentration of a metabolite in the sample.

The fifth imputation method is from the random forest algorithm. The method starts by imputing the median value of a predictor variable within each class present. This completed data set is run through random forest. From this initial run, the proximity of two cases is calculated as all data (including oob) is run down each tree in the forest. During this process every time two cases occupy the same terminal node in a tree, their proximity is increased by one. The cases within the variable that had close proximity to the missing value are used to update the missing value by calculating a weighted average among these observations, where the weights are the proximities. Then this new, complete data set is run through the process again. Literature states

that up to 4-6 iterations should be enough (Brieman, 2003). This idea of updating the missings values each time will give a more accurate fill to be analyzed.

DATA SETS

NCI60

The National Cancer Institute (NCI) has provided a popular metabolomics data set named NCI60. This data set contains 60 cultured cancer cell lines including 9 cancer types (lung, renal, CNS, breast, melanoma, ovarian, colon, leukemia, and prostate). The NCI60 data set was originally distributed to the company Metabolon to obtain metabolite measurements. The final data set provided by Metabolon had a total of 263 metabolites present, several of which were considered unknown. Before the data could be used for analysis, it needed to be cleaned. Each cell had two samples taken and analyzed, with sample 1 on days 1 or 2 and sample 2 on days 3 or 4. Through personal communication with the statistician at Metabolon, the samples collected on days 3 and 4 were not randomized, and therefore are removed from this data set. After removing these observations, the data set then consisted of 57 cell lines.

For illustration purposes, Table 5 displays a partial layout for the NCI60. See that there is a response consisting of "Cancer Type". Each metabolite has a concentration in undeclared units of measurement. Notice the blank cells indicate missing data. This is a normal set-up for a metabolites dataset.

	Metabolites			
Cancer Type	1	2	3	4
renal	108511.3	919117.3	137047.9	
renal	371160.5	355277.4	83873.5	
renal	139468.1	513686	71420.33	338173
renal	145455.5		898870.9	548448.7
renal	41537.86	729172.3		
renal	2443495	520489.7	1199803	573857.9
renal	151743			
prostate	1629358	652242.7	979124.5	561510.6
prostate	3346058	1035335	789694.1	533502.1
ovarian		888605.4	98599.8	541586.6

Table 5. Part of NCI60 Data Set

A Second Metabolomics Dataset

A second metabolomics dataset also created by Metabolon had 58 samples with 317 metabolites measured. This data set has 28 disease cases and 30 non-diseased. This data set, which must remain confidential, was used to confirm results using the NCI60 data set.

Simulations

Simulations of data are created through known processes that setup possible scenarios to be explored and to provide information on the accuracy of different methods. This research uses the data sets described above to help create simulated data. In order to simulate a known response, the original response is stripped away. Then, metabolites that had more than 20% missing were removed. This created an NCI60 data set with 205 metabolites measured for each of the 57 samples. The second metabolomics data set handled this way, created 277 metabolites for each of the 58 samples. By retaining metabolites with more than 80% of observations present, this research hopes to maintain the correlation structure found in a real Metabolomics experiment. However, before simulations can begin, the missing values need to be replaced. Two different methods were used to replace these missing values and create the full data set:

1) random forest algorithm was used to replace missing values in the NCI60 data set, and

2) uinform(0, Θ_i) was used to replace the missing values of the second metabolomics data set.

The resulting two completed data sets were used to simulate responses and explore the various analysis and imputation methods.

In addition to the two real metabolomics data sets, a fully simulated data set was created. The idea was to create a variance-covariance matrix with compound symmetry. The compound symmetry was induced by setting a "block" of a p by p matrix, in this case p=300. Each "block" will be of size 10 by 10 with 1 along the diagonal and a common covariance, a, for all pairings of the 10 variables. In the first block of 10 variables (1 to 10) a=0.9, a=0.8 for the second block of 10 (11 to 20), continuing down by 0.1 for each set of 10 thereafter until a=0.0 for the last 10 (291 to 300). The resulting simulated data set has 60 samples and 300 metabolites, a dataset having similar dimensions to the NCI60 dataset.

Simulation Code

The methods of t-tests, Random Forests and Partial Least Squares have contributed packages available in R software 2.0.4. Each package was loaded into the R session before the simulation was run. The simulation code first randomly chooses three variables or metabolites to generate the response. Using Boolean logic and the median value of the chosen three variables, the code classifies each observation into a response. For example, let variables 3, 7, and 12 of a metabolomics dataset be chosen to create the response, with median values 59, 75, and 125, respectively. Then the Boolean logic code to simulate the response would be

if (variable 1 >= 59) and (variable 2 > =75) then response=0 else

if (variable 1 >= 59) and (variable 2 < 75) then response=1 else

if (variable 1 < 59) and(variable 3 >= 125) then response=0 else

if (variable 1 < 59) and (variable 3 < 125) then response=1.

For the code used to create these simulations, please see Appendix A. These three metabolites used to generate the response are the important features in the simulation. Since this research is interested in exploring which methods are able to identify the important features, these three metabolites will be sought. Also of interest is which imputation methods are best and how much does each method of imputation affect the analysis methods. In order to evaluate the imputation methods, the complete data set (with simulated response) needs to have some missing values for metabolites. Therefore, "holes" or missing values are created in this data set. The following procedure was used to create missing values for each metabolite:

1) A random draw from a beta(1.5, 5.1) is obtained, let this value be q.

2) Let q*100 be the "percent missing".

3) Observations lower than the q*100 percentile are changed to missing.

The above algorithm is repeated for each metabolite. The beta(1.5,5.1) distribution was not an arbitrarily chosen distribution, as this was the approximate distribution of missing values for the original NCI60 data set (57 samples with 278 metabolites measured).

RESULTS

Imputation Bias

Within each run of the simulation, the bias was calculated between the "full" data set and the imputed data set. By definition, bias is the expected value of the estimator minus the true value (Berger, 2002). The sum of the total bias was calculated as the summation over all cells.

This bias value for each imputation method was compared amongst the imputation methods. Here, the highest and lowest bias of each imputation method within each run was determined.

The results of 1000 runs using the complete NCI60 dataset set as discussed in the simulated section are presented in Table 6. Out of 1000 runs, 508 of these for the method of half minimum imputation had the smallest bias seen. The remaining 492 runs had a smallest bias for the uniform imputation method. The highest bias came from Random Forest for all 1000 simulations. This is not surprising since random forest will most likely impute values above the minimum (when these values should be below the minimum).

Imputation	Minimum Bias	Maximum Bias
Random Forest	0	1000
Minimum	0	0
Half Minimum	508	0
Uniform	492	0
0	0	0

Table 6. Bias for NCI60

Results from 1000 runs using the completed second metabolomics dataset set are presented in Table 7. Out of 1000 runs, 509 of these had the smallest bias for the half minimum imputation method. The remaining 491 runs had the smallest bias for the uniform imputation method. The highest bias came from Random Forest for all 1000 simulations. The results from the two metabolomics data sets are consistent.

Imputation	Minimum Bias	Maximum Bias
Random Forest	0	1000
Minimum	0	0
Half Minimum	509	0
Uniform	491	0
0	0	0

Table 7. Bias for Second Metabolomics Data Set

Using the third fully simulated dataset set in this simulation code with 1000 runs generated different results. These results are presented in Table 8. All 1000 runs had the smallest bias as the minimum imputation. The highest bias came from imputing 0 for all 1000 simulations.

Imputation	Minimum Bias	Maximum Bias
Random Forest	0	0
Minimum	1000	0
Half Minimum	0	0
Uniform	0	0
0	0	1000

Table 8. Bias for Simulated Data Set

These results are different from the previous results. However, since the previous results use actual metabolomics data sets, credibility will be placed on previous results.

Feature Selection

Feature selection is an important concept in metabolomics experiments. Feature selection can help identify important biomarkers associated with a given disease. Pathways that are affected by disease can also be identified through this process of feature selection. This research is interested in which methods are able to capture important features. The definition of captured features for each of the analysis methods is given below.

1) An important feature is considered captured by the random forest algorithm if it is in the top 10 features selected. In other words, it is among the 10 highest variable importance scores (defined by the Gini criterion)

2) A feature is considered to be captured by the t-test if the Bonferroni corrected p-value is below 0.05.

3) A feature is considered important for the Partial Least Squares if its loading is among the top 10 highest in the first component. Each method was evaluated using the five imputation methods (RF, minimum, half minimum, 0, and uniform).

Within the simulation code, each simulation run calculated how many of the three variables used in the response simulation were found to be significant in both the original (complete) data and imputed data. A counter in the simulation code summed up how many of the simulations picked up the same or more important values than that of the original. This number was summed for each imputation type over all of the simulation runs and divided by 1000, with the result reported as run percent captured. A second counter summed how many of the important variables were found in each run. Then for each imputation method this number was divided by 3000, result with the reported as total percent captured.

Table 9 show the results when the metabolites from NCI60 imputed data set were used in this code. The evaluation of using Random Forest for feature selection showed that the imputation methods of the 0, the minimum, half of the minimum, and uniform$(0,\Theta_0)$ provided better results than using just the Random Forest imputation. This is noted by the run percent captured of 95% (Table 9). A value of 95% indicates that 950 of the simulated runs found that these imputation methods picked up the same importance variables when compared to the original data set run through Random Forest.

In Table 9, looking at the Random Forest method for total percent captured, there was at least a 57% recovery for the important variables with these same imputation methods. This would mean that out of the potential 3 variables found for each run, there were a total of 3000 that could have been picked up and 57% would provided about 1700 of these.

Analysis Method	Imputation Method	Run Percent Captured (%)	Total Percent Captured (%)
t-test	Original (no imputation)	27.5	21.9
RF	Original (no imputation)	-	57.5
RF	RF	28.5	26.7
RF	minimum	95.8	57.3
RF	half minimum	96.0	57.6
RF	uniform	96.5	57.4
RF	0	96.5	57.6
t-test	RF	17.3	15.1
t-test	minimum	24.9	20.1
t-test	half minimum	27.4	22.8
t-test	uniform	26.4	21.5
t-test	0	27.8	22.5

Table 9. Percent Capture with NCI60

Looking at the RF with the original data set (Table 9) results showed about the same at 57%. Random Forests using the NCI60 data set showed that only 285 of the 1000 simulations provided feature selection at least the same as the original RF run (Table 9), with only about 28% of the 3000 potentials recovered.

Provided in Table 9 are also the results evaluating the t-test with the use of the NCI60. For all imputation methods, percent captured results that feature selection with the t-tests provide a range of about 17-27% of the 1000 simulations picking up the important variables at least as good as the original. Also, total percent captured (Table 9) provided 15-22% of the 3000 potential values being picked up. When looking at the original t-test results, these values are close to the same.

Looking at run and total percent captured (Table 10) results with the second metabolomics data set provided similarities with that of the NCI60. When looking at the t-test, there were lower values shown than what was noted for the NCI60. There seems to be only 23 runs noted for the RF imputation that select at least the same amount of features. There were

about 5-11% of the potential 3000 variables picked up when analyzing with t-test from Random Forest and minimum imputed datasets.

Analysis Method	Imputation Method	Run Percent Captured (%)	Total Percent Captured (%)
t-test	Original (no imputation)	6.80	10.3
RF	Original (no imputation)	-	62.4
RF	RF	22.4	27.3
RF	minimum	97.6	62.1
RF	half minimum	97.6	62.1
RF	uniform	97.6	62.2
RF	0	97.5	62.2
t-test	RF	2.30	5.40
t-test	minimum	5.60	9.50
t-test	half minimum	7.50	11.1
t-test	uniform	7.30	10.4
t-test	0	7.40	11.4

Table 10. Percent Captured for Second Metabolomics Data Set

Using the third fully simulated data set provided almost the same results as was found with the NCI60. Table 11 present the results for run and total percent captured. RF for feature selection with using the 0, minimum, half minimum, and uniform imputation again provide greater than 95% of the 1000 runs detecting at least the same number of important features as the original.

Analysis Method	Imputation Method	Run Percent Captured (%)	Total Percent Captured (%)
t-test	Original (no imputation)	30.2	26.5
RF	Original (no imputation)	-	60.7
RF	RF	33.4	32.2
RF	minimum	96.2	60.5
RF	half minimum	96.5	60.8
RF	uniform	97.0	60.7
RF	0	96.2	60.4
t-test	RF	18.9	18.5
t-test	minimum	27.6	24.6
t-test	half minimum	29.9	26.7
t-test	uniform	28.3	25.2
t-test	0	27.0	24.6

Table 11. Total Percent Captured with Third Fully Simulated Data

Partial Least Squares

In a similar fashion, the Partial Least Squares was also evaluated in the simulation code, the only difference from the first simulation code coming from the calculation of how many runs found important features. An important feature is considered to be selected if the loadings on the first component were in the top ten.

Looking at the NCI60, there seems to be a distinction between the RF imputation and the others. In Table 12, "Total Percent" calculates the percent of the 3000 variables that were picked up in the 1000 runs. RF imputation only picked up 27.7% of the 3000 potential variables (total=0.277*3000=834) from the total of 1000 simulations. The others pick up close to about the same number of variables, from 39.2% to 41.9%. This shows that at least 1200 variables were picked up. "Run Percent" calculates what percent of the simulations were able to pick up at least one important feature in each simulation. RF again had a lower percentage (about 65% or 650 runs) of runs that picked up at least one important variable (Table 12). The other imputation methods are showing at least 80% of the runs had at least one variable. Since the Total Percent of these imputation methods had the highest number of important variables picked up, it would be

expected that their Run Percent should be amongst the highest. However, at least 1200 of the all the 3000 variables were picked up, implying that the Run percent will be between 400 (1200/3) and 1000 (1200/1) runs of the 1000 simulations. These numbers were seen to be about 800 variables picked up, therefore showing that some runs had more than one important variable pick up.

Imputation Method	Run Percent (%)	Total Percent (%)
RF	64.8	27.7
minimum	80.3	39.2
half minimum	83.4	41.6
uniform	82.3	41.0
0	84.3	41.9

Table 12. Partial Least Squares with NCI60

Using PLS with the second metabolomics data set provided very similar results to that of the NCI60. First, Random Forest still provided the smaller percentages (Run Percent=64.8%, Total Percent=27.7%, Table 13) for variable pick up. Run Percent ranged 84.7% to 86.3% for the other imputation methods, so at least 847 runs selected one simulated important variable for feature selection. A range of 40.0% to 41.3% for Total Percent showed that at least 1200 of the 3000 potential variables were selected out of the 1000 runs. Both Run and Total Percent provided very similar results to that of the NCI60 data with PLS analysis (Tables 12 and 13).

Imputation Method	Run Percent (%)	Total Percent (%)
RF	64.9	27.7
minimum	84.8	40.0
half minimum	86.3	40.9
uniform	84.7	40.6
0	85.4	41.3

Table 13. Partial Least Squares with Second Data Set

Using PLS with the third fully simulated metabolomics data set provided very similar results to that of the NCI60 and that of the second. First, Random Forest still provided the smaller percentages (Run Percent=74.2%, Total Percent=34.1%, Table 14) for variable pick up. Run Percent ranged 92.2% to 95.7% for the other imputation methods, so at least 920 runs selected one simulated important variable for feature selection. A range of 46.3% to 48.9% for Total Percent showed that at least 1389 of the 3000 potential variables were selected out of the 1000 runs. Both Run and Total Percent provided very similar results to that of the NCI60 and second metabolomics data sets with PLS analysis (Tables 12 and 13 and 14), noticed with a small 10% increase with Run Percent.

Imputation Method	Run Percent (%)	Total Percent (%)
RF	74.2	34.1
minimum	94.3	48.9
half minimum	95.7	48.9
uniform	94.3	48.5
0	92.2	46.3

Table 14. Partial Least Squares with Third Fully Simulated Data Set

Robust Singular Value Decomposition

The Robust Singular Value Decomposition of a data set provides information on clustering of rows and columns. Here the rows and columns represent samples and metabolites, respectively. Since rSVD can only look at one data set at a time, a few simulated data sets were run through rSVD. The code for rSVD was provided by Dr. Stanley Young from NISS and is in the JMP software. Each of the three data sets created will only simulate one response, but the analysis will still explore the different imputation methods. One thing about rSVD is the possibility of looking at missing values, so one of the data sets will retain the simulated missing values. Will the simulated responses have an effect on the clustering of metabolites? This is just

an attempt to see the impact of simulated response on the clustering of the metabolites. Comparisons will be made about the sorted first right eigenvector (Comp1).

The first dataset was the NCI60. Presented in Table 15 are the 10 highest values for the first right eigenvector for each imputation method. Notice that the same metabolites are identified with missing values and when using the minimum to impute. There appears to be a strong relationship between these metabolites.

Missing Values		Impute Minimum		Impute Half-Min	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
V161	0.262427	V161	0.269622	V161	0.270221
V31	0.249504	V31	0.253065	V31	0.255577
V173	0.242984	V192	0.244937	V192	0.254801
V192	0.232193	V173	0.244038	V8	0.236233
V142	0.215664	V142	0.22644	V142	0.235475
V104	0.211302	V8	0.225391	V13	0.233155
V8	0.208600	V13	0.223807	V173	0.230005
V13	0.206619	V104	0.20276	V67	0.206273
V62	0.205463	V67	0.197092	V55	0.192780
V194	0.201058	V194	0.192252	V104	0.191363
Impute Uniform		Impute 0		Impute RF	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
V161	0.270365	V161	0.270633	V161	0.262219
V192	0.254452	V192	0.264619	V173	0.256779
V31	0.254337	V31	0.259171	V31	0.254602
V8	0.236706	V8	0.247198	V192	0.233941
V142	0.234648	V142	0.244968	V142	0.215193
V13	0.233729	V13	0.242299	V104	0.212756
V173	0.228493	V173	0.216453	V13	0.207979
V67	0.206565	V67	0.215473	V8	0.207721
V104	0.193329	V30	0.19391	V62	0.201616
V55	0.190884	V55	0.192525	V194	0.199726

Table 15. RSVD Results for NCI60

The second data set was the second metabolomics data set. Table 16 shows the 10 highest values for the first right eigenvector for each of the different imputation methods for the data matrix amongst the simulated response.

Missing Values		Impute Minimum		Impute Half-Min	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
M30	0.049440	M122	0.135547	M122	0.138987
M275	0.021997	M109	0.128046	M109	0.130017
M59	0.018849	M192	0.122947	M192	0.125655
M38	0.016725	M67	0.120653	M67	0.124348
M95	0.015258	M177	0.114866	M127	0.116177
M14	0.013834	M127	0.112768	M158	0.115836
M282	0.01296	M179	0.11253	M177	0.115329
M243	0.010815	M158	0.110736	M179	0.115172
M304	0.010228	M223	0.110218	M87	0.113469
M172	0.010044	M87	0.110101	M189	0.109845
Impute Uniform		Impute 0		Impute RF	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
M122	0.139159	M122	0.142247	M291	0.192542
M109	0.130966	M109	0.131829	M192	0.158818
M192	0.125358	M192	0.12822	M280	0.1238
M67	0.124405	M67	0.127876	M122	0.12347
M127	0.11586	M158	0.121042	M179	0.121424
M158	0.115647	M127	0.119302	M109	0.117915
M177	0.115094	M179	0.118015	M223	0.117426
M179	0.114316	M87	0.116555	M127	0.110212
M87	0.113446	M177	0.115652	M158	0.1091
M189	0.109803	M189	0.112854	M177	0.10841

Table 16. RSVD Results for Second Metabolomic Data

The last data set was the third fully simulated data. Table 17 shows the 10 highest values for the first right eigenvector for each of the different imputation methods for the data matrix amongst the simulated response.

Missing Values		Impute Minimum		Impute Half-Min	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
X93	0.08435	X273	0.07576	X92	0.0742
X91	0.08161	X93	0.07525	X91	0.07383
X31	0.07587	X91	0.07399	X273	0.07109
X83	0.07486	X92	0.07012	X33	0.07046
X237	0.07408	X31	0.06892	X31	0.06833
X32	0.07332	X33	0.06716	X32	0.06811
X110	0.07298	X271	0.06697	X271	0.0679
X88	0.07279	X272	0.06688	X115	0.0679
X92	0.07279	X32	0.06683	X93	0.06776
X118	0.07277	X115	0.06595	X272	0.06737
Impute Uniform		Impute 0		Impute RF	
Variable	Comp 1	Variable	Comp 1	Variable	Comp 1
X91	0.07416	X92	0.0785	X273	0.08001
X92	0.0738	X33	0.07395	X93	0.07859
X273	0.07003	X91	0.07349	X91	0.07494
X33	0.06982	X120	0.0707	X31	0.07013
X32	0.06836	X235	0.07067	X237	0.06971
X115	0.06791	X227	0.07052	X83	0.06913
X272	0.06781	X115	0.06985	X271	0.06829
X31	0.067634	X32	0.069463	X59	0.06826
X93	0.06755	X271	0.068751	X118	0.067963
X271	0.067132	X96	0.067725	X32	0.067555

Table 17. RSVD Results for Third Fully Simulated

In looking at the three data sets under different imputation methods, a pattern has been noticed to occur. In these data sets, the ordered 10 highest metabolites have a reoccurrence no matter the imputation type.

DISCUSSION AND CONCLUSIONS

Is there a best imputation or analysis method? The first goal of this research was to evaluate these methods in order to set forth a way to further pursue feature selection of a metabolomics data set. By looking at the evaluations of the imputation methods, the bias shows that there were some conflicting results. For the two real metabolomics data sets the half-

minimum and uniform imputation was seen to be the lowest. Thinking about what these two imputations are doing, it makes sense that half the minimum value would have the smallest distance between results. For the uniform imputation, this is still a value below the minimum meaning that there is a possibility that the bias may be low. Random Forest imputation gave the maximum bias of these two data sets. The fully simulated data set had bias results showing imputing the minimum gave the lowest bias, while imputing 0 gave the largest bias. However, there is more confidence in the two real metabolomics data used when looking at these results.

Looking at the evaluations of the analysis methods for important metabolites showed that using Random Forest might be the best method in order to pick up these variables. All three data sets used with imputation methods of 0, minimum, half minimum, and uniform imputation provided greater than 90% of the 1000 simulations to pick up at least the same as when looking at the original data set.

Partial Least Squares did show favorable results for the NCI60 simulated data when not using Random Forests for the imputation. Imputations of the minimum, half-minimum, 0, or uniform$(0,\Theta_i)$ did show to select at least one of the important variables for a little over 80% of the 1000 runs of the simulations. In the aspect of a "Total" count for the 3000 possible important variables to be selected, only less than half were considered important in their simulation run. The other two data sets provided similar information to that of the NCI60 simulations.

The rSVD of a data matrix did not appear to be much different amongst the imputation methods. Looking at the first 5 metabolites featured in Table 15, there are some similarities to pick out. First, the NCI60 data set shows that there is a strong occurrence of metabolites V8, V31, V142, V161, and V192. Looking at results for the other two data sets show metabolites stay clustered regardless of the response or imputation method.

The NCI60 dataset was explored further to see what metabolites would be featured when looking at two cancer types. This version of the NCI60 had 243 metabolites with no more than 50% missing values present. Since the bias of the uniform(0, Θ_i) was seen to give the lowest for all of the imputation methods, this was also used in the 50% or less missing in the NCI60. Melanoma skin cancer cells are suppose to act more similar to normal healthy cells versus any other cancer type mentioned in this dataset. Therefore, melanoma was thought of as a control for the following analyses. The chosen cancer type was lung cancer, both cancer types provided 9 samples each. So the following analysis will be conducted on a dataset consisting of 18 cell samples and 243 metabolites.

The Random Forests analysis method results for feature selection are provided in Figure 4. The most important metabolite was seen to be metabolite 67. Notice in the plot metabolites 59, 124, 186, 229, and 213 seem also to be important metabolites. Of the remaining top 30, these seem to be fairly significant, but noticing a decrease as going down the plot.

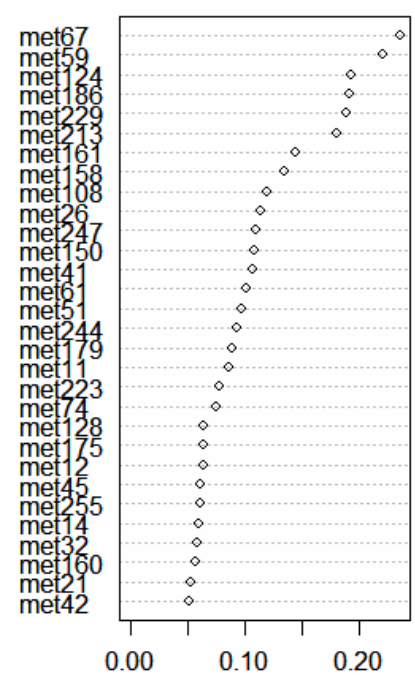

Figure 4. Random Forest NCI60 Importance Plot

These melanoma and lung cancer cells looked at with this method as a way to maybe evaluate PLS with the Random Forest results. Presented in Table 18 are the 10 highest important variables selected in this data. Notice that in comparison to the Random Forest, there are some similar metabolites selected. Most noticeably metabolite 67 is again seen as of highest importance, which was that of RF. Looking at both Figure 4 and Table 20, the other importance metabolites are to be selected by PLS as in RF (exception of metabolite 221).

PLS
met67
met124
met158
met150
met161
met41
met247
met59
met213
met160
met21
met175
met12
met221

Table 18. NCI60 Important Metabolite by PLS

The results presented have provided an evaluation on some possible methods for feature selection and imputation. To further this research, there is a great need for more real life metabolomics data sets. This research has led a pathway to better understanding the issue of missing data and feature selection.

BIBLIOGRAPHY

Abdi, Herve, (2006), The Method of Least Squares, http://www.utdallas.edu/~herve/abdi-leastsquares06-pretty.pdf, November, 2006.

Beecher, C., (2005), Metabolomics: New Applications, New Science, Innovations in Pharmaceutical Technology, Summer/Autumn, 2005

Beecher, C., Lin, X., Troung, Y., (2004), Learning a complex metabolomic dataset using random forests and support vector machines, KDD, Seattle, WA.

Berger, R., Casella, G., (2002), Statistical Inference, 2nd Ed., Brooks/Cole

Breiman, L., (2001), Random Forest, University of California at Berkeley, January, 2001

Breiman, L., (2003), A Class of Two-eyed Algorithms, SIAM Workshop, May, 2003

Ding, B, Gentleman, R., (2004), Classification Using General Partial Least Squares, Bioconductor Project Working Papers, Year 2004

Friedman, J., Hastie, T., Tibshirani, R., (2001), The Elements of Statistical Learning, Springer, New York, 2001

Gabriel, K. R., Zamir , S., (1979), Lower rank approximation of matrices by least squares with any choice of weights, Technometrics, 21, 489-498

Hawkins, D., Liu, L., Young, S., (2001) Robust Singular Value Decomposition, NISS Technical Report Number 122,December, 2001

Abdi, H., (2003), Partial Least Squares (PLS) Regression, Program in cognition and Neurosciences, Dallas, 2003

Kegg Pathway Database for Amino Sugar Pathway, http://www.genome.jp/kegg/ pathway.html, September, 2006

National Cancer Institute (2005), Frontiers in Metabolomics for Cancer Research (meeting abstracts), Rockville, MD

Schmidt, S., (2004), Metabolomics: What's Happening Downstream of DNA, Environmental Health Perspectives, 112:7, May, 2004

Tobias, R., (2006), An introduction to partial least squares regression, SAS Institute Inc, Cary, NC

APPENDIX

A. Simulation and Feature Selection Code

```
SAMSI<-function(X,n)
{library(randomForest)
testsum1<-matrix(nrow=11,ncol=n)
testsum2<-matrix(nrow=12,ncol=n)
bias.min<-(c(0,0,0,0,0))
bias.max<-(c(0,0,0,0,0))
ttestfun<-function(xx)
{testval<-t.test(xx~y3,var.equal=FALSE)
pvaltest<-testval$p.value
return(pvaltest)}
for (j in 1:n)
{y1<-rep(0,length(X[,1]))
tt<-0
while (tt==0)
{variab<-sample(1:length(X[,1]),3,replace=FALSE)
##select 3 variables to simulate response
for (i in 1:length(X[,1]))
{if  (X[i,variab[1]] > median(X[,variab[1]]) & X[i,variab[2]] > median(X[,variab[2]])) y1[i] <-0
else
 {if (X[i,variab[1]] > median(X[,variab[1]]) & X[i,variab[2]] < median(X[,variab[2]])) y1[i] <-1
else
 {if (X[i,variab[1]] < median(X[,variab[1]]) & X[i,variab[3]] > median(X[,variab[3]])) y1[i] <-0
else
 {if (X[i,variab[1]] < median(X[,variab[1]]) & X[i,variab[3]] < median(X[,variab[3]])) y1[i] <-
1}}}}
if (sum(y1) > 20 & sum(y1) < (length(y1) - 20)) tt <- 3}
y2<-rep(NA,length(X[,1]))
y2<-ifelse(y1==0,"A","B")
##response needs to be categorical for RF
susan.1<-cbind(y2,X)
pp<-sum(y1)/length(X[,1])
susan.rf<-randomForest(y2~.,data=susan.1,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
testsum.orig<-testrf
##see how original data picks up features--rf
```

```
testsum2[12,j]<-testrf
y3<-ifelse(y1==0,2,y1)
pvaltest<-apply(X,2,ttestfun)
testsum<-0
 if (pvaltest[variab[1]] < (0.05/length(X[1,])))  testsum<-testsum + 1
 if (pvaltest[variab[2]] < (0.05/length(X[1,]))) testsum<-testsum + 1
 if (pvaltest[variab[3]] < (0.05/length(X[1,]))) testsum<-testsum + 1
if (testsum >= testsum.orig) testsum1[1,j]<-1 else testsum1[1,j]<-0
testsum2[1,j]<-testsum
## see how original data picks up features--t-test
newX<-X
##create missing values
for (i in 1:length(X[1,]))
{u0<-X[,i]
missing.val<-rbeta(1,1.5,7.1)
m0<-quantile(u0,probs=missing.val)
newX[u0 < m0,i]<-NA}
newX2<-data.frame(newX,y2)
##*****Imputing data*****##
susan.2Impute<-rfImpute(y2~.,data=newX2)## susan.2Impute is rf imputation
susan.3Impute<-newX2  ## susan.3Impute is min imputation
susan.4Impute<-newX2  ## susan.4Impute is half-min
susan.5Impute<-newX2  ## susan.5Impute is uniform(0,min)
susan.6Impute<-newX2  ## susan.6Impute is 0
for (i in 1:(length(susan.3Impute[1,])-1))
{while (NA %in% susan.3Impute[,i])
{temp<-match(NA,susan.3Impute[,i])
mintemp<-min(susan.3Impute[,i],na.rm=TRUE)
susan.3Impute[temp,i]<- mintemp
susan.4Impute[temp,i]<-0.5*mintemp
susan.5Impute[temp,i]<-runif(1,0,mintemp)
susan.6Impute[temp,i]<-0}}
##*****Bias*****##
dummy<-susan.2Impute[1,]
bias_rf<-susan.2Impute[,2:length(newX2[1,])]-X
#RF (y2,x-matrix), others (x-matrix,y2)
sumb_rf<-sum(bias_rf)
sumb_rf
bias_min<-susan.3Impute[,1:(length(dummy)-1)]-X
sumb_min<-sum(bias_min)
sumb_min
bias_hmin<-susan.4Impute[,1:(length(dummy)-1)]-X
sumb_hmin<-sum(bias_hmin)
sumb_hmin
bias_uni<-susan.5Impute[,1:(length(dummy)-1)]-X
sumb_uni<-sum(bias_uni)
```

```
sumb_uni
bias_0<-susan.6Impute[,1:(length(dummy)-1)]-X
sumb_0<-sum(bias_0)
sumb_0
emvect<-c(sumb_rf, sumb_min, sumb_hmin, sumb_uni, sumb_0)
em.min<-match(min(abs(emvect)), abs(emvect))
em.max<-match(max(abs(emvect)), abs(emvect))
bias.min[em.min]<-bias.min[em.min]+1
bias.max[em.max]<-bias.max[em.max]+1
##*****Random Forest*****##
susan.rf<-randomForest(y2~.,data=susan.2Impute,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
if (testrf >= testsum.orig) testsum1[2,j]<-1 else testsum1[2,j]<-0
testsum2[2,j]<-testrf
susan.rf<-randomForest(y2~.,data=susan.3Impute,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
if (testrf >= testsum.orig) testsum1[3,j]<-1 else testsum1[3,j]<-0
testsum2[3,j]<-testrf
susan.rf<-randomForest(y2~.,data=susan.4Impute,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
if (testrf >= testsum.orig) testsum1[4,j]<-1 else testsum1[4,j]<-0
testsum2[4,j]<-testrf
susan.rf<-randomForest(y2~.,data=susan.5Impute,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
```

```
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
if (testrf >= testsum.orig) testsum1[5,j]<-1 else testsum1[5,j]<-0
testsum2[5,j]<-testrf
susan.rf<-randomForest(y2~.,data=susan.6Impute,importance=TRUE,replace =
FALSE,sampsize=c(20,20),proximity=TRUE,ntree=10000)
order.rf<-order(susan.rf$importance[,4])
high.rf<-order.rf[(length(X[1,])-9):length(X[1,])]
testrf<-0
if (variab[1] %in% high.rf) testrf<-testrf + 1
if (variab[2] %in% high.rf) testrf<-testrf + 1
if (variab[3] %in% high.rf) testrf<-testrf + 1
if (testrf >= testsum.orig) testsum1[6,j]<-1 else testsum1[6,j]<-0
testsum2[6,j]<-testrf
#*****t-test*****#
onlyX2<-susan.2Impute[,2:(length(susan.2Impute[1,]))]
onlyX3<-susan.3Impute[,1:(length(susan.3Impute[1,])-1)]
onlyX4<-susan.4Impute[,1:(length(susan.4Impute[1,])-1)]
onlyX5<-susan.5Impute[,1:(length(susan.5Impute[1,])-1)]
onlyX6<-susan.6Impute[,1:(length(susan.6Impute[1,])-1)]
pvaltest<-apply(onlyX2,2,ttestfun)
testsum<-0
 if (pvaltest[variab[1]] < (0.05/length(onlyX2[1,])))  testsum<-testsum +1
 if (pvaltest[variab[2]] < (0.05/length(onlyX2[1,]))) testsum<-testsum +1
 if (pvaltest[variab[3]] < (0.05/length(onlyX2[1,]))) testsum<-testsum +1
if (testsum >= testsum.orig) testsum1[7,j]<-1 else testsum1[7,j]<-0
testsum2[7,j]<-testsum
pvaltest<-apply(onlyX3,2,ttestfun)
testsum<-0
if (pvaltest[variab[1]] < (0.05/length(onlyX3[1,])))  testsum<-testsum +1
 if (pvaltest[variab[2]] < (0.05/length(onlyX3[1,]))) testsum<-testsum +1
 if (pvaltest[variab[3]] < (0.05/length(onlyX3[1,]))) testsum<-testsum +1
if (testsum >= testsum.orig) testsum1[8,j]<-1 else testsum1[8,j]<-0
testsum2[8,j]<-testsum
pvaltest<-apply(onlyX4,2,ttestfun)
testsum<-0
if (pvaltest[variab[1]] < (0.05/length(onlyX4[1,])))  testsum<-testsum +1
 if (pvaltest[variab[2]] < (0.05/length(onlyX4[1,]))) testsum<-testsum +1
 if (pvaltest[variab[3]] < (0.05/length(onlyX4[1,]))) testsum<-testsum +1
if (testsum >= testsum.orig) testsum1[9,j]<-1 else testsum1[9,j]<-0
testsum2[9,j]<- testsum
pvaltest<-apply(onlyX5,2,ttestfun)
testsum<-0
if (pvaltest[variab[1]] < (0.05/length(onlyX5[1,])))  testsum<-testsum +1
 if (pvaltest[variab[2]] < (0.05/length(onlyX5[1,]))) testsum<-testsum +1
 if (pvaltest[variab[3]] < (0.05/length(onlyX5[1,]))) testsum<-testsum +1
```

```
if (testsum >= testsum.orig) testsum1[10,j]<-1 else testsum1[10,j]<-0
testsum2[10,j]<- testsum
pvaltest<-apply(onlyX6,2,ttestfun)
testsum<-0
if (pvaltest[variab[1]] < (0.05/length(onlyX6[1,])))  testsum<-testsum +1
 if (pvaltest[variab[2]] < (0.05/length(onlyX6[1,]))) testsum<-testsum +1
 if (pvaltest[variab[3]] < (0.05/length(onlyX6[1,]))) testsum<-testsum +1
if (testsum >= testsum.orig) testsum1[11,j]<-1 else testsum1[11,j]<-0
testsum2[11,j]<-testsum
}
counts.sum<-apply(testsum1,1,sum)
counts.sum2<-apply(testsum2,1,sum)
names<-paste(1:11)
names[1]<-"original t-test"
names[2]<-"RF-Imputed RF"
names[3]<-"RF-Imputed min"
names[4]<-"RF-Imputed halfmin"
names[5]<-"RF-Imputed unif"
names[6]<-"RF-Imputed 0"
names[7]<-"ttest-Imputed RF"
names[8]<-"ttest-Imputed min"
names[9]<-"ttest-Imputed halfmin"
names[10]<-"ttest-Imputed unif"
names[11]<-"ttest-Imputed 0"
names2<-paste(1:12)
names2<-names
names2[12]<-"RF original"
names3<-names[2:6]
names4<-names[2:6]
pp.sum<-counts.sum/n
pp2.sum<-counts.sum2/(3*n)
out.final1<-cbind(names,pp.sum)
out.final2<-cbind(names2,pp2.sum)
out.final3<-cbind(names3,bias.min)
out.final4<-cbind(names4,bias.max)
return(list(out.final1,out.final2,out.final3,out.final4))}
#*****Partial Least Squares*****#
SAMSI<-function(X,n){
pls.comp1<-(c(0,0,0,0,0))
pls.comp2<-(c(0,0,0,0,0))
for (j in 1:n){
tt<-0
while (tt==0)
{variab<-sample(1:length(X[1,]),3,replace=FALSE)
##select 3 variables to simulate response
y1<-rep(0,length(X[,1]))
```

```
for (i in 1:length(X[,1]))
{if  (X[i,variab[1]] > median(X[,variab[1]]) & X[i,variab[2]] > median(X[,variab[2]])) y1[i] <-0
else
 {if (X[i,variab[1]] > median(X[,variab[1]]) & X[i,variab[2]] < median(X[,variab[2]])) y1[i] <-1
else
 {if (X[i,variab[1]] < median(X[,variab[1]]) & X[i,variab[3]] > median(X[,variab[3]])) y1[i] <-0
else
 {if (X[i,variab[1]] < median(X[,variab[1]]) & X[i,variab[3]] < median(X[,variab[3]])) y1[i] <-
1}}}}
if (sum(y1) > 20 & sum(y1) < (length(y1) - 20)) tt <- 3}
y2<-ifelse(y1==0,"A","B")
##response needs to be categorical for RF and RSVD
#make missing data set:
newX<-X
miss<-rep(0,length(X[1,]))
##create missing values
for (i in 1:length(X[1,]))
#add a miss variable
{u0<-X[,i]
missing.val<-rbeta(1,1.5,7.1)
miss[i]<-missing.val
m0<-quantile(u0,probs=missing.val)
newX[u0 < m0,i]<-NA}
newX2<-data.frame(newX,y2)
miss#Can impute with RF
library("randomForest")
susan.2Impute<-rfImpute(y2~.,data=newX2)
susan.3Impute<-newX2  ## susan.3Impute is min imputation
susan.4Impute<-newX2  ## susan.4Impute is half-min
susan.5Impute<-newX2  ## susan.5Impute is uniform(0,min)
susan.6Impute<-newX2  ## susan.6Impute is 0
for (i in 1:(length(susan.3Impute[1,])-1))
{while (NA %in% susan.3Impute[,i])
{temp<-match(NA,susan.3Impute[,i])
mintemp<-min(susan.3Impute[,i],na.rm=TRUE)
susan.3Impute[temp,i]<- mintemp
susan.4Impute[temp,i]<-0.5*mintemp
susan.5Impute[temp,i]<-runif(1,0,mintemp)
susan.6Impute[temp,i]<-0}}
```